

# On the Tree Augmentation Problem

Zeev Nutov

The Open University of Israel. [nutov@openu.ac.il](mailto:nutov@openu.ac.il).

**Abstract.** In the TREE AUGMENTATION problem we are given a tree  $T = (V, F)$  and an additional set  $E \subseteq V \times V$  of edges, called “links”, with positive integer costs  $\{c_e : e \in E\}$ . The goal is to augment  $T$  by a minimum cost set of links  $J \subseteq E$  such that  $T \cup J$  is 2-edge-connected. Let  $M$  denote the maximum cost of a link. Recently, Adjashvili [1] introduced a novel LP for the problem and used it to break the natural 2-approximation barrier for instances when  $M$  is a constant. Specifically, his algorithm computes a  $1.96418 + \epsilon$  approximate solution in time  $n^{(M/\epsilon^2)^{O(1)}}$ . Using a simpler LP, we achieve ratio  $\frac{12}{7} + \epsilon$  in time  $2^{O(M/\epsilon^2)}$ . In particular, this gives ratio better than 2 for logarithmic costs, and not only for constant costs as in other work.

## 1 Introduction

We consider the following problem:

TREE AUGMENTATION

*Input:* A tree  $T = (V, F)$  and an additional set  $E \subseteq V \times V$  of edges, called **links**, with positive integer costs  $c = \{c_e : e \in E\}$ .

*Output:* A minimum cost link set  $J \subseteq E$  such that  $T \cup J$  is 2-edge-connected.

The problem was studied extensively, c.f. [10,14,3,19,7,4,17,5,16,2,15]. For a long time the best known ratio for the problem was 2 for arbitrary costs [10] and 1.5 for unit costs [7,16]. It is also known that the integrality gap of a standard LP-relaxation for the problem, so called CUT-LP, is at most 2 [10] and at least 1.5 [4]. Several LP and SDP relaxations were introduced to show that the algorithm in [7,8,16] achieves ratio better than 2 w.r.t. to these relaxations, c.f. [2,15]. For additional algorithms with ratio better than 2 for restricted versions of the problem see [5,17].

Let  $M$  denote the maximum link cost. Recently, Adjashvili [1] introduced a novel LP for the problem – so called the  $k$ -BUNDLE-LP, and used it to break the natural 2-approximation barrier for instances when  $M$  is bounded by a constant. To introduce this result we need some definitions.

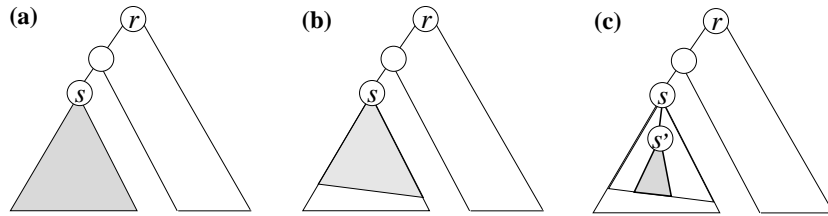
An equivalent formulation of the TREE AUGMENTATION problem is as follows. Let  $T_{uv}$  denote the unique  $uv$ -path in  $T$ . We say that a **link**  $uv$  **covers an edge**  $f$  if  $f \in T_{uv}$ . Then  $T \cup J$  is 2-edge-connected if and only if  $J$  covers  $T$ . For an edge set  $B \subseteq F$  let  $\text{cov}(B)$  denote the set of links in  $E$  that cover some  $f \in B$ , and  $\tau(B)$  the minimum cost of a link set in  $E$  that covers  $B$ . The standard LP

for the problem which we call the CUT-LP seeks to minimize  $c^\top x = \sum_{e \in E} c_e x_e$  over the CUT-POLYHEDRON  $\Pi_k^{\text{Cut}} \subseteq \mathbb{R}^E$  defined by the constraints:

$$\begin{aligned} \sum_{e \in \text{cov}(f)} x_e &\geq 1 & \forall f \in F \\ x_e &\geq 0 & \forall e \in E \end{aligned}$$

The  $k$ -BUNDLE-LP of Adjashvili [1] adds over the standard CUT-LP the constraints  $\sum_{e \in \text{cov}(B)} c_e x_e \geq \tau(B)$  for any forest  $B$  in  $T$  that has at most  $k$  leaves, where  $k = \Theta(M/\epsilon^2)$ . The algorithm of [1] computes a  $1.96418 + \epsilon$  approximate solution w.r.t. the  $k$ -BUNDLE-LP in time  $n^{k^{O(1)}}$ . For unit costs, Adjashvili designed a modification of the algorithm that achieves ratio  $5/3 + \epsilon$ .

Here we observe that it is sufficient to consider just certain subtrees of  $T$  instead of forests. Root  $T$  at some node  $r$ . The choice of  $r$  defines an ancestor/descendant relation on  $V$ . The **leaves of  $T$**  are the nodes in  $V \setminus \{r\}$  that have no descendants. For any subtree  $S$  of  $T$ , the node  $s$  of  $S$  closest to  $r$  is the root of  $S$ , and the pair  $S, s$  is called a **rooted subtree** of  $T, r$ ; we will not mention the roots of trees if they are clear from the context. We say that  $S$  is a **complete rooted subtree** if it contains all descendants of  $s$  in  $T$ , and a **full rooted subtree** if for any non-leaf node  $v$  of  $S$  the children of  $v$  in  $S$  and  $T$  coincide; see Fig. 1 (a) and (b), respectively. A **branch of  $S$** , or a **branch hanging on  $s$** , is a rooted subtree  $B$  of  $S$  induced by the root  $s$  of  $S$  and the descendants in  $S$  of some child  $s'$  of  $s$ ; see Fig. 1 (c). We say that a subtree  $B$  of  $T$  is a **branch** if it is a branch of a full rooted subtree, or if it is a full rooted subtree with root  $r$ . Equivalently, a branch is a union of a full rooted subtree and the parent edge of this subtree.



**Fig. 1.** (a) a complete rooted subtree; (b) a full rooted subtree; (c) a branch of a full rooted subtree.

For  $k \geq 3$  let  $\mathcal{B}_k$  denote the set of all branches in  $T$  with less than  $k$  leaves. The  $k$ -BRANCH-LP seeks to minimize  $c^\top x = \sum_{e \in E} c_e x_e$  over the  $k$ -BRANCH-

POLYHEDRON  $\Pi_k^{Br} \subseteq \mathbb{R}^E$  defined by the constraints:

$$\begin{aligned} \sum_{e \in \text{cov}(f)} x_e &\geq 1 & \forall f \in F \\ \sum_{e \in \text{cov}(B)} c_e x_e &\geq \tau(B) & \forall B \in \mathcal{B}_k \\ x_e &\geq 0 & \forall e \in E \end{aligned}$$

The constraints of the  $k$ -BRANCH-LP is a subset of constraints of the  $k$ -BUNDLE-LP of Adjashvili [1], hence the  $k$ -BRANCH-LP is both more compact and its optimal value is no larger than that of the  $k$ -BUNDLE-LP. The main result of this paper is:

**Theorem 1.** *For any  $1 \leq \lambda \leq k-1$ , there exists a  $4^k \cdot \text{poly}(n)$  time algorithm that given a TREE AUGMENTATION instance computes a solution of cost at most  $\rho + \frac{8}{3} \frac{\lambda M}{k-\lambda} + \frac{2}{\lambda}$  times the optimal value of the  $k$ -BRANCH-LP, where where  $\rho = \frac{12}{7}$  for arbitrary costs and  $\rho = 1.6$  for unit costs.*

For a given  $\epsilon$ , choosing properly  $\lambda = \Theta(1/\epsilon)$  and  $k = \Theta(M/\epsilon^2)$  gives ratio  $\rho + \epsilon$  in time  $2^{O(M/\epsilon^2)} \cdot \text{poly}(n)$ .

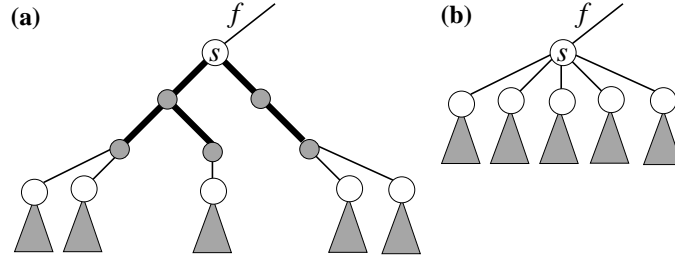
We note that recently Fiorini et. al. [9] augmented the  $k$ -BUNDLE LP of [1] by additional constraints – so called  $\{0, \frac{1}{2}\}$ -Chvátal-Gomory Cuts, to achieve ratio  $1.5 + \epsilon$  in  $n^{(M/\epsilon^2)^{O(1)}}$  time, thus matching the best known ratio for unit costs [16]. Our work, done independently, shows that already the  $k$ -BUNDLE LP has integrality gap closer to 1.5 than to 2. Our version of the algorithm of [1] is also simpler than the one in [9]. In fact, combining our approach with [9] enables to achieve ratio  $1.5 + \epsilon$  in  $2^{O(M/\epsilon^2)} \cdot \text{poly}(n)$  time. Note that this is a substantial improvement, as it allows to achieve ratio better than 2 for logarithmic costs, and not only for constant costs.

In the rest of this section we briefly describe our algorithm, which is a modification of the algorithm of Adjashvili [1]. We emphasize the differences. We use the  $k$ -BRANCH-LP instead of the  $k$ -BUNDLE-LP of [1]. But, unlike [1], we do not solve our LP at the beginning, but start with an optimal solution to the CUT-LP. We design an algorithm (see Algorithm 1) that either returns a solution within the stated bound or returns a  $k$ -branch-constraint violated by our current solution  $x$ ; we show that this can be done in time  $4^k \cdot \text{poly}(n)$ , rather than in time  $n^{k^{O(1)}}$  as in [1]. This gives a  $4^k \cdot \text{poly}(n)$  time separation oracle for the  $k$ -BRANCH-LP (if a violated  $k$ -branch constraint is found). Since the ellipsoid algorithm has a polynomial number of calls to the separation oracle, after a polynomial number of iterations, there will be an iteration when no  $k$ -branch constraint violated by  $x$  will be found. At this iteration the algorithm will find a solution of cost at most  $(\rho + \epsilon)c^T x$ , where  $\rho$  is as in Theorem 1 and  $x$  is an optimal solution to the LP formed by the cut-constraints and the  $k$ -branch-constraints found during the algorithm.

In the algorithm, we repeatedly take a certain complete rooted subtree  $S$  of  $T$ , and either find a  $k$ -branch-constraint violated by some subtree of  $S$ , or

a “cheap” cover  $J_S$  of  $S$ ; in the latter case, we add  $J_S$  to our partial solution, contract  $S$ , and iterate. When covering  $S$ , we replace every link  $uv$  with  $u \in S$  and  $v \notin S$  by two links  $us$  and  $sv$ . This makes the problems of covering  $S$  and  $T \setminus S$  independent, but increases the fractional cost  $\sum_{e \in E} c_e x_e$  of  $x$ . This increase is bounded  $\sum_{e \in \text{cov}(f)} c_e x_e$ , where  $f$  is the parent edge of  $S$ . If we choose  $S$  such that the quotient of  $\sum_{e \in \text{cov}(f)} c_e x_e$  over the fractional cost of links with both endnodes in  $S$  is small, then this invokes a small loss in the ratio. We call an edge  $f \in F$   $\lambda$ -**thin** if  $x(\text{cov}(f)) \leq \lambda$ , and  $f$  is  $\lambda$ -**thick** otherwise. We say that a complete rooted subtree  $S$  of  $T$  is a  $(k, \lambda)$ -**subtree** if  $S$  has at least  $k$  leaves and if either the parent edge  $f$  of the root  $s$  of  $S$  is  $\lambda$ -thin or  $s = r$ . For  $\lambda = \Theta(1/\epsilon)$  and  $k = \Theta(M/\epsilon^2)$  we choose  $S$  to be an *inclusionwise minimal*  $(k, \lambda)$ -subtree. Then  $\sum_{e \in \text{cov}(f)} c_e x_e = \Theta(M/\epsilon)$ , the fractional cost of links with both endnodes in  $S$  is at least  $\frac{k-\lambda}{2} = \Theta(M/\epsilon^2)$ , and the above quotient is  $O(\epsilon)$ .

Now let us focus on the problem of covering such  $S$ , see Fig. 2(a), where the  $\lambda$ -thick edges are shown by bolded lines. We contract the inclusionwise maximal subtree containing  $s$  that consists of  $\lambda$ -thick edges, see Fig. 2(b). It is shown in [1] that all  $\lambda$ -thick edges can be covered by cost  $\frac{2}{\lambda} c^\top x = \epsilon c^\top x$  (see Lemma 1), so we postpone covering these edges to the end of the algorithm. Now, after the contraction, every branch  $B$  hanging on  $s$  has less than  $k$  leaves, by the minimality of  $S$ , hence it has a corresponding constraint in the  $k$ -BRANCH-LP.



**Fig. 2.** Branches hanging on  $s$  after contracting the inclusionwise maximal subtree containing  $s$  that consists of  $\lambda$ -thick edges.

A link  $uv$  is called a **cross-link** if  $u$  and  $v$  belong to distinct branches hanging on  $s$ , and  $uv$  is an **in-link** otherwise. As in [1], we choose the better outcome of two procedures.

The first procedure is identical to the one in [1], but we show that it can be implemented in time  $4^k \cdot \text{poly}(n)$ . We compute an *optimal* cover  $J_B$  of each branch  $B$  hanging on  $s$ . If for some branch  $B$  we get  $x(\text{cov}(B)) < \tau(B)$ , then a  $k$ -branch constraint violated by  $x$  is found. Else, the union of the covers  $J_B$  computed is a cover of  $S$  of cost  $2C^{\text{cr}} + C^{\text{in}}$ , where  $C^{\text{cr}}$  and  $C^{\text{in}}$  denote the fractional cost of cross-links and in-links, respectively.

In the second procedure we replace each in-link  $uv$  by two **up-links**  $ua$  and  $va$  of the same cost as  $uv$ , where  $a$  is the least common ancestor of  $uv$ . Then we

compute an extreme point solution of the CUT-LP in the modified instance; this part differs from the one in [1]. We show that in the case when every in-link is an up-link, such a solution is always *half integral*, see Lemma 4. We round such a solution to an integral solution within a factor of  $4/3$  using the algorithm of [3], and get a solution of cost  $\frac{4}{3}(2C^{\text{in}} + C^{\text{cr}})$ ; in the case of unit costs we improve this to  $2C^{\text{in}} + \frac{4}{3}C^{\text{cr}}$ .

## 2 The main algorithm (Theorem 1)

To prove Theorem 1 in the next section we prove the following.

**Theorem 2.** *Suppose that we are given an instance of TREE AUGMENTATION and  $x \in \Pi^{\text{Cut}}$  such that any proper complete rooted subtree of the input tree has less than  $k$  leaves. Then there exists a  $4^k \cdot \text{poly}(n)$  time algorithm that either finds a  $k$ -branch inequality violated by  $x$ , or computes a solution of cost  $\leq \rho \sum_{e \in E \setminus R} c_e x_e + \frac{4}{3} \sum_{e \in R} c_e x_e$ , where  $\rho = \frac{12}{7}$  for arbitrary costs and  $\rho = 1.6$  for unit costs, and  $R$  is the set of edges in  $E$  incident to the root.*

In the rest of this section we will show that Theorem 2 implies Theorem 1. Recall that given  $x \in \mathbb{R}^E$  we say that an edge  $f \in F$  is  $\lambda$ -**thin** if  $x(\text{cov}(f)) \leq \lambda$ , and  $f$  is  $\lambda$ -**thick** otherwise. We need the following simple lemma.

**Lemma 1 ([1]).** *There exists a polynomial time algorithm that given  $x \in \Pi^{\text{Cut}}$  and a set  $F' \subseteq F$  of  $\lambda$ -thick edges computes a cover  $J'$  of  $F'$  of cost  $\leq \frac{2}{\lambda} \cdot c^\top x$ .*

*Proof.* Since all edges in  $F'$  are  $\lambda$ -thick,  $x/\lambda$  is a feasible solution to the CUT-LP for covering  $F'$ . Thus any polynomial time algorithm that computes a solution  $J'$  of cost at most 2 times the optimal value of the CUT-LP for covering  $F'$  has the desired property. There are several such algorithms, see [10,11,13].

We say that a complete rooted subtree  $S$  of  $T$  is a  $(k, \lambda)$ -**subtree** if  $S$  has at least  $k$  leaves and if either the parent edge  $f$  of the root  $s$  of  $S$  is  $\lambda$ -thin or  $s = r$ . Given such  $S$  with  $s \neq r$  we apply the following natural operation. **Cutting a link**  $e = uv$  **at a node**  $s$  means that if  $s$  is an internal node of  $T_{uv}$  then we set

$$x_{us} \leftarrow x_{us} + x_e \quad x_{sv} \leftarrow x_{sv} + x_e \quad x_e \leftarrow 0$$

This operation keeps  $x$  a feasible solution to the CUT-LP or to the  $k$ -BRANCH-LP and increases  $c^\top x$  by at most  $c_e x_e$ . **Cutting**  $S$  means sequentially cutting at  $s$  every link  $e = uv$  such that  $s$  is an internal node of  $T_{uv}$ .

We denote by  $T/S$  the tree obtained from  $T$  by repeatedly contracting every edge of  $S$ . Note that this defines a new TREE AUGMENTATION instance, where contraction of an edge  $uv$  leads to shrinking  $u, v$  into a single node in the graph  $(V, E)$  of links.

The algorithm as in Theorem 1 works in iterations. Each iteration starts with a feasible solution  $x$  to a subset of constraints of the  $k$ -BRANCH-LP that includes all the constraints of the CUT-LP. At each iteration the algorithm

either finds a  $k$ -branch inequality violated by  $x$ , or returns a solution of cost at most  $\left(\rho + \frac{8}{3} \frac{\lambda M}{k-\lambda} + \frac{2}{\lambda}\right) c^T x$  and terminates. In parallel, we apply the ellipsoid algorithm, for the case that a  $k$ -branch inequality violated by  $x$  is found and added to the maintained subset of constraints. The following procedure describes the main iteration of the algorithm.

---

**Algorithm 1:** ITERATION( $T = (V, F), x, c, k, r, \lambda$ )

---

```

1  $J \leftarrow \emptyset, F' \leftarrow \emptyset$ 
2 while  $J$  does not cover  $T$  do
3   cut an inclusionwise minimal  $(k, \lambda)$ -subtree  $S$  of  $T$ 
4   let  $S'$  be the edge-set of the inclusionwise maximal subtree of  $S$  that
     contains the root  $s$  of  $S$  and has only  $\lambda$ -thick edges (possibly  $S' = \emptyset$ )
5   apply the algorithm from Theorem 2 on  $S/S'$ 
6   if the algorithm returns a  $k$ -branch constraint violated by  $x$  then
     return this inequality and STOP
7   else, if the algorithm returns a cover  $J_S$  of  $S/S'$  do
      $F' \leftarrow F' \cup S', J \leftarrow J \cup J_S, T \leftarrow T/S$ 
8 compute a cover  $J'$  of  $F'$  using the algorithm from Lemma 1
9 return  $J \cup J'$ 

```

---

Note that any proper complete rooted subtree of the tree  $S/S'$  considered at step 5 of the algorithm has less than  $k$  leaves and thus Theorem 2 indeed applies. Also, at step 7 the edges in  $F'$  are all  $\lambda$ -thick and thus Lemma 1 applies. We will now analyze the performance of the algorithm assuming than no  $k$ -branch inequality violated by  $x$  was found. Let  $\delta(S)$  denote the set of links with exactly one endnode in  $S$  and  $\gamma(S)$  the set of links with both endnodes in  $S$ . Let  $f$  be the parent edge of  $S$ . Since  $f$  is  $\lambda$ -thin

$$\sum_{e \in \text{cov}(f)} c_e x_e \leq \sum_{e \in \text{cov}(f)} M x_e \leq M \cdot x(\text{cov}(f)) \leq M \lambda .$$

Since  $x(\delta(v)) \geq 1$  for every leaf  $v$  of  $S$ ,  $x_e \geq 1$  for every  $e \in E$ , and since  $S$  is a  $(k, \lambda)$ -subtree

$$\sum_{e \in \gamma(S)} c_e x_e \geq \sum_{e \in \gamma(S)} x_e \geq \frac{1}{2} \left( \sum_{v \in S \setminus \{s\}} x(\delta(v)) - \lambda \right) \geq \frac{k - \lambda}{2} .$$

Since after  $S$  is cut, all the cut links are incident to  $s$  and since  $\rho \geq 4/3$  we get

$$c(J_S) \leq \rho \sum_{e \in E \setminus R} c_e x_e + \frac{4}{3} \sum_{e \in R} c_e x_e \leq \rho \sum_{e \in \gamma(S)} c_e x_e + \frac{4}{3} M \lambda \leq \left( \rho + \frac{8}{3} \frac{M \lambda}{k - \lambda} \right) \sum_{e \in \gamma(S)} c_e x_e$$

From this it follows by induction that at the end of the algorithm  $c(J)$  is at most  $\rho + \frac{8}{3} \frac{\lambda M}{k - \lambda} + \frac{2}{\lambda}$  times  $c^T x$ . Thus it only remains to prove Theorem 2, which we will do in the next section.

### 3 Proof of Theorem 2

Assume that we are given an instance of TREE AUGMENTATION as in Theorem 2. It is known that TREE AUGMENTATION instances when  $T$  is a path can be solved in polynomial time. This allows us to assume that the graph  $(V, E)$  of links is a complete graph and that  $c_{uv} = \tau(T_{uv})$  for all  $u, v \in V$ . Let us say that a link  $uv \in E$  is:

- a **cross-link** if  $r$  is an internal node of  $S_{uv}$ ;
- an **in-link** if  $r$  does not belong to  $S_{uv}$ ;
- an  **$r$ -link** if  $r = u$  or  $r = v$ ;
- an **up-link** if one of  $u, v$  is an ancestor of the other.

For a set  $E' \subseteq E$  of links the  $E'$ -**up vector** of  $x$  is obtained from  $x$  as follows: for every non-up link  $e = uv \in E'$  increase  $x_{ua}$  and  $x_{va}$  by  $x_e$  and then reset  $x_e$  to 0, where  $a$  is the least common ancestor of  $u$  and  $v$ . The **fractional cost** of a set  $B$  of links w.r.t.  $c$  and  $x$  is defined by  $\sum_{e \in B} c_e x_e$ . Let  $C_x^{\text{in}}$ ,  $C_x^{\text{cr}}$ , and  $C_x^r$  denote the fractional cost of in-links, cross-links, and  $r$ -links, respectively, w.r.t.  $c$  and  $x$ . We fix some  $x^* \in \Pi^{\text{Cut}}$  and denote by  $C^{\text{in}}$ ,  $C^{\text{cr}}$ , and  $C^r$  the fractional cost of in-links, cross-links, and  $r$ -links, respectively, w.r.t.  $c$  and  $x^*$ . We give two rounding procedures, given in Lemmas 2 and 3.

**Lemma 2.** *There exists a  $4^k \cdot \text{poly}(n)$  time algorithm that either finds a  $k$ -branch inequality violated by  $x^*$ , or returns an integral solution of cost at most  $C^{\text{in}} + 2C^{\text{cr}} + C^r$ .*

*Proof.* Let  $\mathcal{B}$  be the set of branches hanging on  $r$ . For every  $B \in \mathcal{B}$  compute an optimal solution  $J_B$ . If for some  $B \in \mathcal{B}$  we have  $x^*(\text{cov}(B)) < \tau(B)$  then a  $k$ -branch inequality violated by  $x^*$  is found. Else, the algorithm returns the union  $J = \bigcup_{B \in \mathcal{B}} J_B$  of the computed edge sets. We show that  $c(J) \leq C^{\text{in}} + 2C^{\text{cr}} + C^r$ . Let  $E^{\text{cr}}$  be the set of cross-links and let  $x'$  be the  $E^{\text{cr}}$ -up vector of  $x^*$ . Then  $x'$  satisfies the  $k$ -branch inequalities of the branches in  $\mathcal{B}$ , has value  $C^{\text{in}} + 2C^{\text{cr}} + C^r$ , and  $x'_e = 0$  for every cross-link  $e$ ; in particular, every link  $e$  with  $x'_e > 0$  belongs to at most one set  $\text{cov}(B)$ . Thus  $c(J) \leq \sum_{e \in E} c_e x'_e \leq C^{\text{in}} + 2C^{\text{cr}} + C^r$ , as required.

It remains to show that an optimal solution in each branch of  $r$  can be computed in time  $4^k \cdot \text{poly}(n)$ . More generally, we will show that TREE AUGMENTATION instances with  $k$  leaves can be solved optimally within this time bound. Recall that we may assume that the graph  $(V, E)$  of links is a complete graph and that  $c_{uv} = \tau(T_{uv})$  for all  $u, v \in V$ . We claim that then we can assume that  $T$  has no node  $v$  with  $\deg_T(v) = 2$ . This is a well known reduction, c.f. [18]. In more details, we show that any solution  $J$  can be converted into a solution of no greater costs that has no link incident to  $v$ , and thus  $v$  can be “shortcut”. If  $J$  has links  $uv, vw$  then it is easy to see that  $(J \setminus \{uv, vw\}) \cup \{uw\}$  is also a feasible solution, of cost at most  $c(J)$ . Applying this operation repeatedly we may assume that  $\deg_J(v) \leq 1$ . If  $\deg_J(v) = 0$ , we are done. Suppose that  $J$  has a unique link  $e = vw$  incident to  $v$ . Let  $vu$  and  $vu'$  be the two edges of  $T$  incident to  $v$ , where assume that  $vu'$  is not covered by  $e$ . Then there is a link  $e' \in J$  that

covers  $vu'$ . Since  $e'$  is not incident to  $v$ , it must be that  $e'$  covers  $vu$ . Replacing  $e$  by the link  $wu'$  gives a feasible solution without increasing the cost.

Consequently, we reduce our instance to an equivalent instance with at most  $2k - 1$  tree edges. Now recall that TREE AUGMENTATION is a particular case of the MIN-COST SET-COVER problem, where the set  $F$  of edges of  $T$  are the elements and  $\{T_e : e \in E\}$  are the sets. It is known that the MIN-COST SET-COVER problem can be solved in  $2^n \cdot \text{poly}(n)$  time via dynamic programming, where  $n$  is the number of elements [6]. Thus our reduced TREE AUGMENTATION instance can be solved in  $2^{2k-1} \cdot \text{poly}(n) \leq 4^k \cdot \text{poly}(n)$  time.

For the the second rounding procedure Adjashvili [1] proved that for any  $\lambda > 1$  one can compute in polynomial time an integral solution of cost at most  $2\lambda C^{\text{in}} + \frac{4}{3} \frac{\lambda}{\lambda-1} C^{\text{cr}}$ . We prove:

**Lemma 3.** *There exists a polynomial time algorithm that computes a solution of cost  $\frac{4}{3}(2C^{\text{in}} + C^{\text{cr}} + C^r)$ , and a solution of size  $2C^{\text{in}} + \frac{4}{3}C^{\text{cr}} + C^r$  in the case of unit costs.*

Consider the case of arbitrary bounded costs. If  $C^{\text{in}} \geq \frac{2}{5}C^{\text{cr}}$  we use the rounding procedure from Lemma 2 and the rounding procedure from Lemma 3 otherwise. In both cases we get  $c(J) \leq \frac{12}{7}(C^{\text{in}} + C^{\text{cr}}) + \frac{4}{3}C^r$ .

In the case of unit costs, if  $C^{\text{in}} \geq \frac{2}{3}C^{\text{cr}}$  we use the rounding procedure from Lemma 2, and the procedure from Lemma 3 otherwise. In both cases we get  $c(J) \leq 1.6(C^{\text{in}} + C^{\text{cr}}) + C^r$ .

In the rest of this section we prove Lemma 3. The proof relies on properties of extreme points of the CUT-POLYHEDRON given in Lemmas 4 and 5; these properties are of independent interest. Note that although  $\Pi^{\text{Cut}}$  is not a polytope, the CUT-LP always has an optimal solution  $x$  that is an **extreme point** or a **basic feasible solution** of  $\Pi^{\text{Cut}}$ . Geometrically, this means that  $x$  is not a convex combination of other points in  $\Pi^{\text{Cut}}$ ; algebraically this means that there exists a set of  $|E|$  inequalities in the system defining  $\Pi^{\text{Cut}}$  such that  $x$  is the unique solution for the corresponding linear equations system. These definitions are known to be equivalent and we will use both of them.

A set family  $\mathcal{L}$  is **laminar** if any two sets in the family are either disjoint or one contains the other. Note that TREE AUGMENTATION is equivalent to the problem of covering the laminar family of the node sets of the full rooted proper subtrees of  $T$ , where a link covers a node set  $A$  if it has exactly one endnode in  $A$ . In particular, note that the constraint  $\sum_{e \in \text{cov}(f)} x_e \geq 1$  is equivalent to the constraint  $x(\delta(A)) \geq 1$  where  $A$  is the node set of the full rooted subtree with parent edge  $f$ . Let us say that an instance of TREE AUGMENTATION is **star shaped** if every in-link in  $E$  is an up-link.

**Lemma 4.** *Let  $x$  be an extreme point of  $\Pi^{\text{Cut}}$  with  $0 < x_e < 1$  for all  $e \in E$  on a star shaped instance of TREE AUGMENTATION. Then  $x_e = 1/2$  for all  $e \in E$ .*

*Proof.* Let  $\mathcal{L}$  be a laminar family such that  $x$  is the unique solution to the equation system  $\{x(\delta(A)) = 1 : A \in \mathcal{L}\}$ , where  $|\mathcal{L}| = |E|$ .



*Claim.* Every  $A \in \mathcal{L}$  has at most 2 children in  $\mathcal{L}$  and  $|\delta(A)| = 2$ .

*Proof.* For every  $uv \in E$  put one token at  $u$  and one token at  $v$ . The total number of tokens is  $2|E|$ . Let  $t(A)$  denote the number of tokens placed at nodes in  $A$  that belong to no child of  $A$ . Since  $\mathcal{L}$  is laminar,  $\sum_{A \in \mathcal{L}} t(A) \leq 2|E|$ . We will show that for every  $A \in \mathcal{L}$  the following holds:

- (i)  $t(A) \geq ch(A)$ , where  $ch(A)$  denotes the number of children of  $A$ .
- (ii)  $t(A) \geq |\delta(A)|$  if  $ch(A) = 0$ , or if  $|\delta(C)| = 2$  holds for every child  $C$  of  $A$ .
- (iii)  $t(A) \geq 2$ .

Note that (i) and (ii) imply that  $t(A) \geq 3$  for any inclusionwise minimal set  $A \in \mathcal{L}$  for which the lemma does not hold, giving together with (iii) the contradiction  $\sum_{A \in \mathcal{L}} t(A) > 2|E|$ .

We prove (i). Consider a child  $C$  of  $A$ . Let  $E_C = \delta(C) \setminus \delta(A)$  denote the set of links in  $E$  that cover  $C$  but not  $A$ . The assumption that every in-link is an up-link implies that there are no links between the children of  $A$ . Thus every link in  $E_C$  contributes 1 to  $t(A)$ . Moreover,  $E_C \neq \emptyset$  by linear independence and since  $x(\delta(A)) = x(\delta(C)) = 1$ . This implies (i).

We prove (ii). If  $ch(A) = 0$  then  $t(A) = |\delta(A)|$ . Assume that  $ch(A) \geq 1$  and that  $|\delta(C)| = 2$  for every child  $C$  of  $A$ . Let  $E_A$  denote the set of links in  $\delta(A)$  that cover no child of  $A$ . Note that every link in  $E_A$  contributes 1 both to  $\delta(A)$  and to  $t(A)$ . Every link in  $\delta(A)$  is either in  $E_A$  or in  $\delta(C) \cap \delta(A)$  for some child  $C$  of  $A$ . Every link in  $E_C$  contributes  $|E_C| \geq 1$  to  $t(A)$ , and  $|\delta(C) \cap \delta(A)| \leq 2 - |E_C| \leq 1$  to  $\delta(A)$ . This implies (ii).

From (i) and (ii) it follows that to prove (iii) it is sufficient to consider the cases  $ch(A) = 1, 2$ . If  $A$  has a unique child  $C$  then since  $x(\delta(A)) = x(\delta(C)) = 1$  and by linear independence,  $E_A \neq \emptyset$ ; thus  $t(A) \geq |E_A \cup E_C| \geq 2$ . If  $A$  has exactly two children, say  $C$  and  $C'$ , then  $t(A) \geq |E_C \cup E_{C'}| \geq 2$ .

From the above claim it follows that the equation system  $\{x(\delta(A)) = 1 : A \in \mathcal{L}\}$  has exactly two variables in each equation. By [12], the solution of such systems is always half-integral.

The algorithm that computes an integral solution of cost  $\frac{4}{3}(2C^{\text{in}} + C^{\text{cr}} + C^r)$  is as follows. We obtain a star shaped instance by removing all non-up in-links and compute an optimal extreme point solution  $x$  to the CUT-LP. By Lemma 4,  $x$  is half-integral. Cheriyan, Jordán & Ravi [3] showed how to round a half-integral solution to the CUT-LP to integral solution within a factor of  $4/3$ . Thus we can compute a solution  $J$  of cost at most  $c(J) \leq \frac{4}{3}c^\top x \leq \frac{4}{3}c^\top x^*$ . We claim that  $c^\top x \leq 2C^{\text{in}} + C^{\text{cr}} + C^r$ . To see this let  $E^{\text{in}}$  be the set of in-links and let  $x'$  be the  $E^{\text{in}}$ -up vector of  $x^*$ . Then  $x'$  is a feasible solution to the CUT-LP of value  $2C^{\text{in}} + C^{\text{cr}} + C^r$ , in the obtained TREE AUGMENTATION instance all non-up in-links removed. But since  $x$  is an optimal solution to the same LP, we have  $c^\top x \leq c^\top x' = 2C^{\text{in}} + C^{\text{cr}} + C^r$ . This concludes the proof of Lemma 3 for the case of arbitrary costs.

For the case of unit costs we prove:

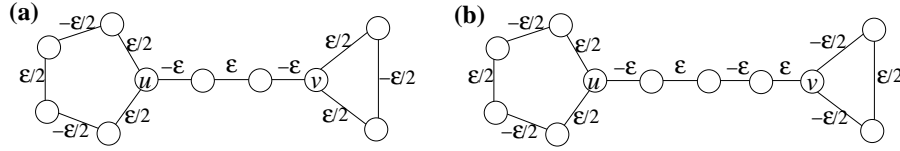
**Lemma 5.** *Let  $a, b \geq 0$  and let  $x$  be an extreme point of the polytope*

$$\Pi = \{x \in \Pi^{Cut} : C_x^{in} = a, C_x^{cr} = b\}.$$

*such that  $x_e > 0$  for every cross-link  $e$ . Then the graph  $(V, E^{cr})$  of cross-links has no even cycles and every its connected component has at most one cycle.*

*Proof.* Let  $Q$  be a cycle in  $E^{cr}$  and let  $\epsilon = \min_{e \in Q} x_e$ . Since  $x_e > 0$  for all  $e \in E^{cr}$ ,  $\epsilon > 0$ . If  $|Q|$  is even, let  $Q', Q''$  be a partition of  $Q$  into two perfect matchings. Let  $z$  be a vector defined by  $z_e = \epsilon$  if  $e \in Q'$ ,  $z_e = -\epsilon$  if  $e \in Q''$ , and  $z_e = 0$  otherwise. By the choice of  $\epsilon$ ,  $x + z, x - z$  are non-negative, and it is not hard to verify that  $x + z, x - z \in \Pi$ . However,  $x = \frac{1}{2}(x + z) + \frac{1}{2}(x - z)$ , contradicting that  $x$  is an extreme point.

Suppose that  $|Q|$  is odd. Let  $u, v$  be nodes on  $Q$ , possibly  $u = v$ . We claim that  $(V, E^{cr} \setminus Q)$  has no  $uv$ -path; this also implies that any two odd cycles in  $(V, E^{cr})$  are node disjoint. Suppose to the contrary that  $(V, E^{cr} \setminus Q)$  has a  $uv$ -path  $P$ . Let  $P'$  and  $P''$  be the two internally disjoint  $uv$ -paths in  $Q$  where  $|P'|$  is odd and  $|P''|$  is even. Then one of  $P \cup P'$  and  $P \cup P''$  is an even cycle, contradicting that  $(V, E^{cr})$  has no even cycles.



**Fig. 3.** Illustration to the proof that no two cycles in  $(V, E^{cr})$  are connected by a path.

Finally, we show that no two cycles in  $(V, E^{cr})$  are connected by a path. Suppose to the contrary that  $(V, E^{cr})$  has a  $uv$ -path  $P$  that connects two distinct cycles  $Q_u$  and  $Q_v$ , see Figure 3. Let  $z$  be defines as Figure 3. By the choice of  $\epsilon$ , each one of the vectors  $x + z$  and  $x - z$  is non-negative, and they are both in  $\Pi$ . However,  $x = \frac{1}{2}(x + z) + \frac{1}{2}(x - z)$ , contradicting that  $x$  is an extreme point.

Note that Lemma 5 implies that extreme points of  $\Pi^{Cut}$  have the property given in the lemma. From Lemma 5 we also get:

**Corollary 1.** *In the case of unit costs there exists a polynomial time algorithm that computes  $x \in \Pi$  such that the graph  $(V, E^{cr})$  of cross links of positive  $x$ -value is a forest and such that  $C_x^{in} = C^{in}$ ,  $C_x^r = C^r$ , and  $C_x^{cr} \leq \frac{4}{3}C^{cr}$ .*

*Proof.* Let  $\Pi$  be as in Lemma 5 where  $a = C^{in}$  and  $b = C^r$  and let  $x$  be an optimal extreme point solution to the LP  $\min\{\sum_{e \in E} x_e : x \in \Pi\}$ . Let  $Q$  be a cycle of cross links and  $e$  the minimum  $x$ -value link in  $Q$ . We update  $x$  by adding  $x_e$  to each of  $x_{e'}, x_{e''}$  and setting  $x_e = 0$ . The increase in the value of  $x$  is at most  $\frac{1}{3} \sum_{e \in Q} x_e$ , and it is not hard to verify that  $x$  remains a feasible solution. In this way we can eliminate all cycles, ending with  $x \in \Pi$  as required.

**Remark.** Corollary 1 holds also for arbitrary costs, but in this case the proof is much more involved. Specifically, we use the following statement, which we do not prove here, since it currently has no application:

*Let  $q \geq 3$  and let  $c_i, x_i \geq 0$  be reals,  $i = 0, \dots, q-1$ . Denote  $a_i = c_{i-1} - c_i + c_{i+1}$  where the indices are modulo  $k$ . Then  $\sum_{i=0}^{k-1} c_i x_i \geq 3 \cdot \min_{0 \leq i \leq k-1} a_i x_i$ .*

Let  $x$  be as in Corollary 1 and let  $x'$  be an  $E^{\text{in}}$ -up vector of  $x$ . Note that  $x' \in \Pi^{\text{Cut}}$ , since  $x \in \Pi^{\text{Cut}}$ . We will show how to compute a solution  $J$  of size  $c(J) \leq x'(E) \leq 2C^{\text{in}} + \frac{4}{3}C^{\text{cr}} + C^{\text{r}}$ . While there exists a pair of links  $e = uv$  and  $e' = u'v'$  such that  $x'_e, x'_{e'} > 0$  and  $T_{u'v'} \subset T_{uv}$  we do  $x'_e \leftarrow x'_e + x'_{e'}$  and  $x'_{e'} \leftarrow 0$ . Then  $x'$  remains a feasible solution to the CUT-LP without changing the value (since we are in the case of unit costs). Hence we may assume that there is no such pair of links. Let  $E'$  be the support of  $x'$ . If every leaf of  $T$  has some cross-link in  $E'$  incident to it, then by the assumption above there are no up-links. In this case, since  $E'$  is a forest,  $x_e \geq 1$  for every  $e \in E'$  and  $E'$  is a solution as required.

Otherwise, there is a leaf  $v$  of  $T$  such that no cross link in  $E'$  is incident to  $v$ . Then there is a unique up-link  $e$  incident to  $v$ , and  $x'_e \geq 1$ . We take such  $e$  into our partial solution, updating  $x'$  and  $E'$  accordingly. Note that some cross links may become  $r$ -links, but no up-link can become a cross-link, and the set of cross-links remains a forest. Applying this as long as such leaf  $v$  exists, we arrive at the previous case, where adding  $E'$  to the partial solution gives a solution as required. This concludes the proof of Lemma 3.

## 4 Conclusions

In this paper we presented an improved algorithm for TREE AUGMENTATION, by modifying the algorithm of Adjashvili [1]. A minor improvement is that the algorithm is slightly simpler, as it avoids a technical discussion on so called “early compound nodes”, see [1]. A more important improvement is in the running time  $-4^k \text{poly}(n)$  instead of  $n^{k^{O(1)}}$ , where  $k = \Theta(M/\epsilon^2)$ . This allows ratio better than 2 also for logarithmic costs, and not only costs bounded by a constant. These two improvements are based, among others, on a more compact LP for the problem. Another important improvement is in the ratio  $-\frac{12}{7} + \epsilon$  instead of  $1.96418 + \epsilon$  in [1]. This algorithm is based on a combinatorial result for star shaped TREE AUGMENTATION instances, when all in-links are up-links. We showed that for star shaped instances, the extreme points of the CUT-POLYHEDRON are half-integral, and thus TREE AUGMENTATION on such instances can be approximated within  $4/3$ . As was mentioned, a related recent result of [9] shows that for star shaped instances, augmenting the cut constraints by  $\{0, \frac{1}{2}\}$ -Chvátal-Gomory Cuts gives a polyhedron with integral extreme points; thus TREE AUGMENTATION on such instances can be solved optimally. Overall we get that star shaped instances behave as TREE AUGMENTATION instances when  $T$  is a star (this is essentially the EDGE-COVER problem): the extreme points of the CUT-LP are half-integral, while augmenting it by  $\{0, \frac{1}{2}\}$ -Chvátal-Gomory Cuts given an integral polyhe-

dron. The description of the  $\{0, \frac{1}{2}\}$ -Chvátal-Gomory Cuts in [9] is somewhat complicated, and we ask whether adding simpler constraints, suggested earlier by the author, gives the same result:

$$x(\text{cov}(A)) \geq \lceil |A|/2 \rceil \quad A \subseteq E, |A| \text{ odd, no 3 edges in } A \text{ lie on the same path}$$

In the case when  $T$  is a star, the last condition on  $A$  is void, and it is known that augmenting the CUT-LP by these constraints gives an integral polyhedron – c.f. [20] where an equivalent EDGE-COVER problem is considered.

**Acknowledgment.** The author thanks Shoni Gilboa, Manor Mendel, Moran Feldman, and Gil Alon for several discussions.

## References

1. D. Adjashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. In *SODA*, pages 2384–2399, 2017.
2. J. Cheriyan and Z. Gao. Approximating (unweighted) tree augmentation via lift-and-project, part II. Manuscript, 2015.
3. J. Cheriyan, T. Jordán, and R. Ravi. On 2-coverings and 2-packing of laminar families. In *ESA*, pages 510–520, 1999.
4. J. Cheriyan, H. Karloff, R. Khandekar, and J. Koenemann. On the integrality ratio for tree augmentation. *Operation Research Letters*, 36(4):399–401, 2008.
5. N. Cohen and Z. Nutov. A  $(1 + \ln 2)$ -approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theoretical Computer Science*, 489-490:67–74, 2013.
6. M. Cygan. Private communication, 2016.
7. G. Even, J. Feldman, G. Kortsarz, and Z. Nutov. A  $3/2$ -approximation for augmenting a connected graph into a two-connected graph. In *APPROX*, pages 90–101, 2001.
8. G. Even, J. Feldman, G. Kortsarz, and Z. Nutov. A 1.8-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Transactions on Algorithms*, 5(2), 2009.
9. S. Fiorini, M. Groß, J. Koenemann, and L. Sanitá. A  $\frac{3}{2}$ -approximation algorithm for tree augmentation via chvátal-gomory cuts. <https://arxiv.org/abs/1702.05567>, Feb 27, 2017.
10. G. N. Frederickson and J. Jájá. Approximation algorithms for several graph augmentation problems. *SIAM J. Computing*, 10:270–283, 1981.
11. M. Goemans, A. Goldberg, S. Plotkin, E. T. D. Shmoys, and D. Williamson. Improved approximation algorithms for network design problems. In *SODA*, pages 223–232, 1994.
12. D. Hochbaum, N. Megiddo, J. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Math. Programming*, 62:6983, 1993.
13. K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
14. S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. *J. of Algorithms*, 14:214–225, 1993.

15. G. Kortsarz and Z. Nutov. LP-relaxations for tree augmentation. In *APPROX-RANDOM*, pages 13:1–13:16, 2016.
16. G. Kortsarz and Z. Nutov. A simplified 1.5-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Trans. on Algorithms*, 12(2):23, 2016.
17. Y. Maduel and Z. Nutov. Covering a laminar family by leaf to leaf links. *Discrete Applied Mathematics*, 158(13):1424–1432, 2010.
18. D. Marx and L. Végh. Fixed-parameter algorithms for minimum-cost edge-connectivity augmentation. *ACM Trans. Algorithms*, 11(4):27, 2015.
19. H. Nagamochi. An approximation for finding a smallest 2-edge connected subgraph containing a specified spanning tree. *Discrete Applied Math.*, 126:83–113, 2003.
20. A. Schrijver. *Combinatorial Optimization, Polyhedra and Efficiency*. Springer-Verlag Berlin, Heidelberg New York, 2004.